Three-dimensional Turbulent Channel

1 Introduction

This case provides a description for three-dimensional turbulent channel ($Re_\tau = 1000$) that is run using a wall-modeled large-eddy simulation (WMLES) paradigm. A constant body force is applied to the domain that drives the flow. The objective of this laboratory case is to understand the one-dimensional exchange-based wall-modeling approach that is common in the WMLES literature, see [1]. Full production runs would require increased mesh resolution - an exercise left to the reader.

2 Domain

The three-dimensional geometry for this tutorial is captured in Figure 1. The top and bottom boundaries are no-slip, while all others are periodic (both spanwise and streamwise). The domain length (streamwise direction, x), width (spanwise direction, z) and height (vertical direction, y) are 2π x $2/3\pi$ x 2, respectively.

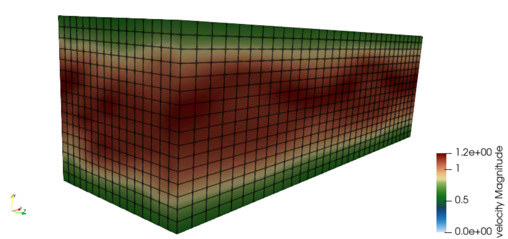

Figure 1: Three-dimensional channel flow.

3 Theory

General details on the WMLES technique can be found in [1], however, in short this is a modeling approach to aliviate the substantial mesh resolution requirement that is required to adequately resolve turbulent boundary layers. For a classic description of mesh requirement scaling, the reader is referred to [2].

For the target $Re_\tau = 1000$ configuration, which is based on the channel half-height, the density (ρ) and viscosity (μ) specification are 1.0 and 5.0×10^{-5}, respectively. The corresponding constant body force is 2.5×10^{-3}.

Rather than resolving the turbulent boundary layer through increased mesh resolution, which would be characterized as a wall-resolved LES (WRLES), the wall shear stress can be modeled using an equilibrium-based approach that exercises the classic law of the wall expression,

$$u^+ = \frac{u_\|}{u_\tau} = \frac{1}{\kappa} \ln\left(y^+\right) + C. \tag{1}$$

Above, $u_\|$ is the parallel velocity, u_τ is the wall friction velocity, κ is a constant (generally, 0.41), y^+ is the dimensionless vertical height, and C is a constant that varies based on the estimated surface roughness (generally 5.1). The dimensionless vertical height is defined by,

$$y^+ = \frac{\rho Y_p u_\tau}{\mu}, \tag{2}$$

where Y_p is the first mesh spacing distance.

Given a parallel velocity, computational first mesh spacing distance and a set of properties (density, viscosity, and surface roughness), the above expression can be solved iteratively to obtain the friction velocity and subsequent wall shear stress, $\tau_w = \rho u_\tau^2$. The integrated wall shear stress forms the wall boundary contribution to the momentum equation that is partitioned based on the vector components of the parallel, or tangential velocity component to the wall.

4 ODE-based Exchange

Assuming an isothermal configuration in which the viscosity and density are constant, the wall shear stress can be modeled using an ordinary-differential equation (ODE) approach [1] where the following underlying one-dimensional equation is solved:

$$\frac{d}{dy}\left[\left(\mu + \mu^t\right)\frac{dU}{dy}\right] = 0.0, \tag{3}$$

where,

$$\mu^t = \kappa \rho \sqrt{\frac{\tau_w}{\rho}} y \left[1 - \exp\left(-y^*/A^+\right)\right]^2, \tag{4}$$

$$y^* = \frac{\rho y u_\tau}{\mu} = \frac{\rho y}{\mu}\sqrt{\frac{\tau_w}{\rho}} = \frac{y\sqrt{\rho \tau_w}}{\mu}. \tag{5}$$

The above constants, A^+ and κ are 17.0 and 0.41, respectively, while the distance y is relative to the wall location and proceeds to the physical exchange location, Δ.

The above set of equations can be discretized over a one-dimensional coordinate system using a finite volume or finite element method. The system mirrors a one-dimensional variable-coefficient diffusion solve since the effective viscosity varies by height. Moreover, the solution procedure is nonlinear as the wall shear stress is a obtained from the iteratively-converging velocity profile, i.e., $\tau_w = \mu \frac{dU}{dy}$. This profile can be computed using a one-sided gradient at the wall, or $y = 0$ location.

4.1 Solution Procedure

A simple node-based iteration solution procedure/algorithm can be represented as follows:

1. Define the one-dimensional set of nodes (or points) that may include variable mesh spacing near the wall for increased wall shear stress calculation accuracy. Here, the wall location may be $y = 0$ and the final point location the exchange distance, $y = \Delta$.

2. Guess a wall shear stress, τ_w and compute the turbulent viscosity from Eq. 4 at each point in the domain.

3. Assemble and solve a matrix system for Eq. 3 whose effective nodal viscosity, i.e., $\mu + \mu^t$, is variable as the values are a function of height and the current wall shear stress. The integration point effective viscosity can be interpolated from the surrounding nodal values. The boundary conditions for velocity are simply: $U = 0$ at $y = 0$, and $U = U^X$ at $y = \Delta$ (here, let U^X be the parallel velocity at the exchange point). For a vertex-based scheme, the rows for each known boundary condition can be modified to enforce the Dirichlet condition.

4. Compute an updated wall shear stress from the recently computed velocity profiled obtained in step 3 above.

5. Determine the error between the previous and most current wall shear stress. If this error is sufficiently small, then the system has been converged. Otherwise, return to step 2 with the latest wall shear stress and repeat. A new matrix system must be computed at each iteration since the matrix coefficients are changing due to an updated effective viscosity.

For more information, the webpage of [3] can be a useful guide, especially if a non-isothermal system is desired to be modeled.

4.2 Example ODE Profile

As an example of the WMLES approach, the untransformed law of the wall dimensionless coordinate velocity profiles for a turbulent flow past an elevated cylinder are depicted in Figure 2. In this figure, both the ODE- and law of the wall, i.e., Eq. 1 are shown. Various mesh points and bias factors are captured along with the converged wall shear stress.

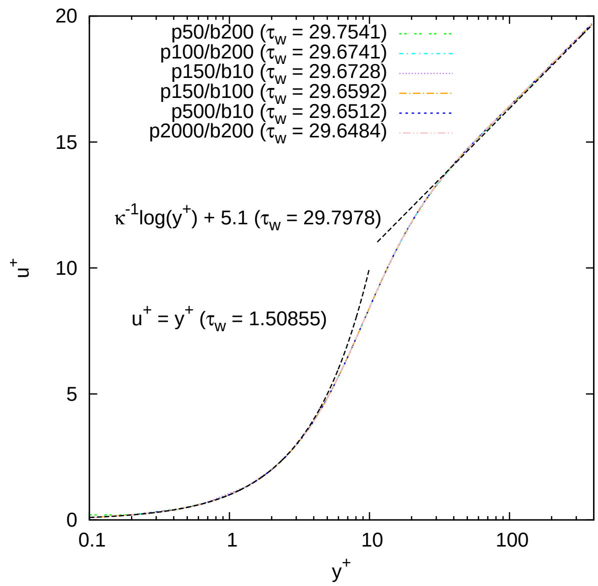

Figure 2: Normalized velocity and normal distance plot showcasing the ODE-based profile and compared to laminar and standard law of the wall. In this configuration the specifications are as follows: parallel exchange velocity, 3.39 m/s; density, 1000 kg/m^3; viscosity, 8.9×10^{-4} $kg/m{-}s$; exchange location outer distance, 0.002 m; constants: A^+ 17 and κ 0.41. In this plot, "p" represents the number of points while "b" represents the bias factor, or the ratio between the last and first mesh spacing.

5 Discussion Points

There are several interesting activities associated with this sample case including the following:

- Replicate Fig. 2 using the specifications provided in this figure for parallel exchange velocity, density, viscosity, exchange distance and constants A^+ and κ. For this step, you can use a discretization of your choice. I prefer using an edge-based vertex-centered (EBVC) finite volume discretization approach.

- For your first experiment, use eleven points that are equally spaced between zero and Δ. Report the computed scaled velocity and distance plot along with the converged wall shear stress. To determine the convergence metric, simply compute the absolute difference between the guessed wall shear stress and the subsequently computed value. When this difference is less than 1.0×10^{-6}, the system has been converged.

- Vary the number of points (from 10, 100, and 200) and report the error in wall shear stress.

- Document a biased, or variable mesh point distance algorithm that would be useful; run this approach with 10, 100, and 200 points and report errors.

- Comment on the linear solver approach that you used to solve this inner system. For this case, a tri-diagonal solve can be extremely efficient, here shown in *c++* using a simple one-dimensional matrix structure, "NaluOneDMatrixSystem", that is size *mSize*, i.e., total number of nodes/points, with matrix entries "Aw", "Ap", and "Ae":

```
void tdma(int mSize, NaluOneDMatrixSystem *ns, double *u) {

  // forward
  for ( int k = 1; k < mSize; ++k ) {
    const double m = ns[k].Aw_/ns[k-1].Ap_;
    ns[k].Ap_ -= m*ns[k-1].Ae_;
    ns[k].rhs_ -= m*ns[k-1].rhs_;
  }

  // backward
  u[mSize-1] = ns[mSize-1].rhs_/ns[mSize-1].Ap_;
  for ( int k = mSize-2; k >= 0; --k ) {
    u[k] = (ns[k].rhs_ - ns[k].Ae_*u[k+1])/ns[k].Ap_;
  }
}
```

Listing 1: This is a tri-diagonal implicit matrix solve routine (also known as the Thomas Algorithm) and is suitable for solving a diagnonal dominant matrix system. For structured three-dimensional applications, alternating sweep directions can be combined with the tri-diagonal solve. In this implementation, the matrix coefficients are modified and are not intended to be re-used for each subsequent iteration.

- Share your code base and methodology for solving this problem in addition to any plots you have created.

- Provide comments on how confident your are that your solution is "correct" and has been coded and/or solved correctly.

- *OPTIONAL, i.e., NOT-REQUIRED*: Run the laboratory regression test and report your findings.

References

[1] Larsson, J., Kawai, S., Bodart, J., and I. Bermejo-Moreno, *Large eddy simulation with modeled wall-stress: recent progress and future directions*, Mech. Engr. Reviews, Vol. 3, 2016.

[2] Chapman, D. R., *Computational aerodynamics development and outlook*, AIAA J., Vol. 3, 1979.

[3] Larsson, J., *Wall-stress model: ODE*, https://wmles.umd.edu/wall-stress-models/wall-model-ode (accessed March 19th, 2024).